

[illegible]

SSSSSSSS	HH	HH	000000	DDDDDDDD	EEEEEEEEEE	VV	VV	CCCCCCCC	LL	UU	UU
SSSSSSSS	HH	HH	000000	DDDDDDDD	EEEEEEEEEE	VV	VV	CCCCCCCC	LL	UU	UU
SS	HH	HH	00	DD	DD	VV	VV	CC	LL	UU	UU
SS	HH	HH	00	DD	DD	VV	VV	CC	LL	UU	UU
SS	HH	HH	00	DD	DD	VV	VV	CC	LL	UU	UU
SS	HH	HH	00	DD	DD	VV	VV	CC	LL	UU	UU
SSSSSS	HHHHHHHHHH	HHHHHHHHHH	00	DD	DD	VV	VV	CC	LL	UU	UU
SSSSSS	HHHHHHHHHH	HHHHHHHHHH	00	DD	DD	VV	VV	CC	LL	UU	UU
SS	HH	HH	00	DD	DD	VV	VV	CC	LL	UU	UU
SS	HH	HH	00	DD	DD	VV	VV	CC	LL	UU	UU
SS	HH	HH	00	DD	DD	VV	VV	CC	LL	UU	UU
SS	HH	HH	00	DD	DD	VV	VV	CC	LL	UU	UU
SSSSSSSS	HH	HH	000000	DDDDDDDD	EEEEEEEEEE	VV	VV	CCCCCCCC	LLLLLLLLLL	UUUUUUUUUU	UUUUUUUUUU
SSSSSSSS	HH	HH	000000	DDDDDDDD	EEEEEEEEEE	VV	VV	CCCCCCCC	LLLLLLLLLL	UUUUUUUUUU	UUUUUUUUUU

```

LL          IIIIII      SSSSSSSS
LL          IIIIII      SSSSSSSS
LL          II         SS
LL          II         SS
LL          II         SS
LL          II         SS
LL          II         SSSSSS
LL          II         SSSSSS
LL          II         SS
LL          II         SS
LL          II         SS
LL          II         SS
LLLLLLLLLLL IIIIIIII   SSSSSSSS
LLLLLLLLLLL IIIIIIII   SSSSSSSS

```



```
1 0001 0 MODULE shodevclu (IDENT = 'V04-000'  
2 0002 0 ADDRESSING_MODE (EXTERNAL = GENERAL)) =  
3 0003 0  
4 0004 1 BEGIN  
5 0005 1  
6 0006 1  
7 0007 1  
8 0008 1  
9 0009 1  
10 0010 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY  
11 0011 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.  
12 0012 1 * ALL RIGHTS RESERVED.  
13 0013 1 *  
14 0014 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  
15 0015 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE  
16 0016 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER  
17 0017 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  
18 0018 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY  
19 0019 1 * TRANSFERRED.  
20 0020 1 *  
21 0021 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE  
22 0022 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT  
23 0023 1 * CORPORATION.  
24 0024 1 *  
25 0025 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS  
26 0026 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.  
27 0027 1 *  
28 0028 1 *****  
29 0029 1  
30 0030 1  
31 0031 1 ++  
32 0032 1  
33 0033 1 FACILITY: SHOW utility  
34 0034 1  
35 0035 1 ABSTRACT:  
36 0036 1 This module contains the routines for finding cluster-wide  
37 0037 1 information about devices by chasing through the lock structures.  
38 0038 1  
39 0039 1 ENVIRONMENT:  
40 0040 1 VAX native, user mode.  
41 0041 1  
42 0042 1 AUTHOR: CW Hobbs CREATION DATE: 19-Mar-1984  
43 0043 1  
44 0044 1 MODIFIED BY:  
45 0045 1  
46 0046 1 V03-005 CWH3005 CW Hobbs 4-May-1984  
47 0047 1 Exclude all null locks from consideration, since the are  
48 0048 1 due to other SHOW DEVICE commands rather than something  
49 0049 1 interesting.  
50 0050 1  
51 0051 1 V03-004 CWH3004 CW Hobbs 13-Apr-1984  
52 0052 1 Remove declaration for a debugging routine.  
53 0053 1  
54 0054 1 V03-003 CWH3003 CW Hobbs 13-Apr-1984  
55 0055 1 Change name for LKISL_REMSYSTEM to LKISL_REMSYSID and LKISB_STATE  
56 0056 1 to LKISB_QUEUE because the definitions changed.  
57 0057 1
```



SHODEVCLU  
V04-000

C 8  
16-Sep-1984 01:34:31  
14-Sep-1984 12:09:25

VAX-11 Bliss-32 V4.0-742  
[CLIUTL.SRC]SHODEVCLU.B32;1

Page 2  
(1)

:	58	0058	1	!
:	59	0059	1	!
:	60	0060	1	!
:	61	0061	1	!
:	62	0062	1	!--

V03-002 CWH3002 CW Hobbs 12-Apr-1984  
Complete work now that a full service \$GETLKI is available



```
64 0063 1 |
65 0064 1 | Include files
66 0065 1 |
67 0066 1 |
68 0067 1 | LIBRARY 'SYSSLIBRARY:LIB';
69 0068 1 | REQUIRE 'SRC$:SHOWDEF';
70 0167 1 | REQUIRE 'SRC$:SHODEVDEF';
71 0458 1 |
72 0459 1 |
73 0460 1 | Table of contents
74 0461 1 |
75 0462 1 | FORWARD ROUTINE
76 0463 1 |     scan_cluster_locks : NOVALUE,
77 0464 1 |     get_lock_info,
78 0465 1 |     get_lock_info_handler;
79 0466 1 |
80 0467 1 | EXTERNAL ROUTINE
81 0468 1 |     show$write_line;
82 0469 1 |
83 0470 1 | EXTERNAL LITERAL
84 0471 1 |     show$_lockerr;
85 0472 1 |
86 0473 1 | EXTERNAL
87 0474 1 |     lck$gl_maxid,
88 0475 1 |     lck$gl_idtbl : REF VECTOR [, LONG],
89 0476 1 |     kernel_accvio : VECTOR [4, LONG];
90 0477 1 |
91 0478 1 |
92 0479 1 | Define a structure for a local buffer used to pass various items from
93 0480 1 | the kernel-mode routine back to the user-mode routine
94 0481 1 |
95 0482 1 | MACRO
96 0483 1 |     lcl_null_lkid      = 0, 0, 32, 0 %,
97 0484 1 |     lcl_lengths       = 4, 0, 32, 0 %,
98 0485 1 |     lcl_ret_length    = 4, 0, 16, 0 %,
99 0486 1 |     lcl_itm_length    = 6, 0, 15, 0 %,
100 0487 1 |     lcl_eng_status    = 8, 0, 32, 0 %,
101 0488 1 |     lcl_val_block     = 12, 0, 0, 0 %;
102 0489 1 |
103 0490 1 | LITERAL
104 0491 1 |     lcl_size = 28;
105 0492 1 |
106 0493 1 |
107 0494 1 | We would like to be able to REQUIRE 'SHRLIB$:MOUDEF.B32', but MOUDEF has a bunch of
108 0495 1 | definitions which conflict with our own definitions. Therefore, we have a copy of the
109 0496 1 | definitions which we need.
110 0497 1 |
111 0498 1 | Define fields within the device allocation lock value block.
112 0499 1 |
113 0500 1 |
114 0501 1 | MACRO
115 0502 1 |     DC_FLAGS          = 0,0,16,0 %,
116 0503 1 |     DC_NOTFIRST_MNT  = 0,0,1,0 %,
117 0504 1 |     DC_FOREIGN       = 0,1,1,0 %,
118 0505 1 |     DC_GROUP         = 0,2,1,0 %,
119 0506 1 |     DC_SYSTEM        = 0,3,1,0 %,
120 0507 1 |     DC_WRITE         = 0,4,1,0 %;
```

SHODEVCLU  
V04-000

E 8  
16-Sep-1984 01:34:31  
14-Sep-1984 12:09:25

VAX-11 BLISS-32 V4.0-742  
[CLIUTL.SRC]SHODEVCLU.B32;1

Page 4  
(2)

:	121	0508	1	DC_NOQUOTA	= 0.5,1.0 %;
:	122	0509	1	DC_OVR_PROT	= 0.6,1.0 %;
:	123	0510	1	DC_OVR_OWNUIC	= 0.7,1.0 %;
:	124	0511	1	DC_NOINTERLOCK	= 0.8,1.0 %;
:	125	0512	1	DC_PROTECTION	= 2.0,16.0 %;
:	126	0513	1	DC_OWNER_UIC	= 4.0,32.0 %;



```
128 0514 1 GLOBAL ROUTINE scan_cluster_locks (scratch : REF $BBLOCK, buffer : REF $BBLOCK) : NOVALUE =
129 0515 2 BEGIN
130 0516
131 0517 ---
132 0518
133 0519 This is a user-mode jacket routine for the lock searches
134 0520
135 0521 Inputs
136 0522 SCRATCH - address of scratch data for this device
137 0523
138 0524 Outputs
139 0525 SCRATCH - some values "adjusted" for cluster-wide information
140 0526 BUFFER - output lock information, for now gets a vector containing
141 0527 the CSIDs of the remote nodes. First longword is the count
142 0528 of the CSIDs, longword CSIDs start at second longword
143 0529
144 0530 ---
145 0531
146 0532 OWN
147 0533 local_csid : INITIAL (0); ! The CSID of the local node
148 0534
149 0535 LOCAL
150 0536 status,
151 0537 lclbuf : $BBLOCK[lcl_size], ! Buffer to receive misc items from kernel call
152 0538 lokbuf : $BBLOCK[1200], ! Lock info buffer for kernel routines (set to MAXBUF minimu
153 0539 arglist : VECTOR[16]; ! CMKRNL and output argument list
154 0540
155 0541 BIND
156 0542 csid_vector = buffer[0,0,32,0] : VECTOR [, LONG],
157 0543 csid_count = buffer[0,0,32,0]; ! Treat first longword as the length field
158 0544
159 0545
160 0546 Zero the csid count field in the users buffer, and remember the local state of the mount bit
161 0547
162 0548 csid_count = 0;
163 0549 scratch[d_v_local_mount] = . $BBLOCK[scratch[d_l_devchar], dev$v_mnt];
164 0550
165 0551
166 0552 Get the CSID of the local node if necessary
167 0553
168 0554 IF .local_csid EQL 0
169 0555 THEN
170 0556 BEGIN
171 0557 arglist[0] = (syi$node_csid^16 OR 4);
172 0558 arglist[1] = local_csid;
173 0559 arglist[2] = arglist[3] = 0;
174 0560 IF NOT (status = $getsyi (itmlst=arglist))
175 0561 THEN
176 0562 SIGNAL_STOP (.status);
177 0563 END;
178 0564
179 0565
180 0566 Call the kernel mode routine, since the device lock is a kernel lock
181 0567
182 0568 arglist[0] = 4; ! Four arguments
183 0569 arglist[1] = scratch; ! Device scratch area
184 0570 2 arglist[2] = lclbuf; ! Local buffer for misc returns
```

```
185 0571 2 arglist[3] = %ALLOCATION(lokbuf);      ! Buffer for $GETLKI to place
186 0572 2 arglist[4] = lokbuf;                ! list of lock item blocks
187 0573
188 0574 IF NOT (status = %CMKRN(LROUTIN = get_lock_info, ARGST = arglist))
189 0575 THEN
190 0576 BEGIN
191 0577     IF .status EQL ss$_accvio
192 0578     THEN SIGNAL(show$_lockerr, 2, .scratch[d_b_devlen], scratch[d_t_device], .status,
193 0579         .kernel_accvio[0], .kernel_accvio[1], .kernel_accvio[2], .kernel_accvio[3], 0)
194 0580     ELSE SIGNAL(show$_lockerr, 2, .scratch[d_b_devlen], scratch[d_t_device], .status);
195 0581     RETURN;
196 0582     END;
197 0583
198 0584
199 0585     ! If there are any remote nodes represented, then update the device scratch area and pass the CSID
200 0586     ! back to the caller.
201 0587
202 0588 INCR k FROM 0 TO .lclbuf[lcl_ret_length]-1 BY lki$length
203 0589 DO
204 0590 BEGIN
205 0591 BIND
206 0592     lki = lokbuf[k,0,32,0] : $BBLOCK;
207 0593
208 0594     ! Only look at non-null locks owned by remote nodes.
209 0595
210 0596     ! null locks      - not interesting, most likely just other show device commands
211 0597     ! local locks    - we can find far more info from the ucb than from the lock
212 0598
213 0599 IF .lki[lki$l_remsysid] NEQ .local_csid
214 0600 AND
215 0601     .lki[lki$b_grmode] NEQ lck$k_nlmode
216 0602 THEN
217 0603 BEGIN
218 0604     csid_count = .csid_count + 1;                ! Bump the count of systems
219 0605     csid_vector[csid_count] = .lki[lki$l_remsysid]; ! Copy the CSID into the vector
220 0606     IF .$BBLOCK[lclbuf[lcl_val_block],dc_notfirst_mnt] ! If mounted on the remote node
221 0607     THEN
222 0608 BEGIN
223 0609     scratch[d_v_remote_mounts] = 1;                ! Remember that it is mounted elsewhere
224 0610     $BBLOCK[scratch[d_l_devchar],dev$v_mnt] = 1;    ! Force the MNT bit on
225 0611     scratch[d_w_mcount] = .scratch[d_w_mcount] + 1; ! Bump the mount count
226 0612     $BBLOCK[scratch[d_l_devchar],dev$v_for] =      ! Set the foreign bit if mounted foreign
227 0613         . $BBLOCK[lclbuf[lcl_val_block],dc_foreign];
228 0614     $BBLOCK[scratch[d_l_devchar],dev$v_sw] =        ! Set the write-locked bit
229 0615         NOT . $BBLOCK[lclbuf[lcl_val_block],dc_write];
230 0616     IF NOT .scratch[d_v_local_mount]                ! If not mounted locally, then set
231 0617     THEN                                           ! a dummy volume name
232 0618         CH$MOVE (12, UPLIT BYTE ('(remote mnt)'), scratch[d_t_volnam]);
233 0619     END;
234 0620 IF .lki[lki$b_grmode] EQL lck$k_exmode                ! If lock mode is exclusive, then the
235 0621 THEN                                           ! device is also allocated
236 0622 BEGIN
237 0623     $BBLOCK[scratch[d_l_devchar],dev$v_all] = 1;    ! Force the ALL bit on
238 0624     scratch[d_v_remote_all] = 1;                    ! Set flag for the print routines
239 0625     END;
240 0626 END;
241 0627 2 END;
```



```
242 0628
243 0629
244 0630
245 0631
246 0632
247 0633
248 0634
249 0635
250 0636
251 0637
252 0638
253 0639
254 0640
255 0641
256 0642
257 0643
258 0644
259 0645
260 0646
261 0647
262 0648
263 0649
264 0650
265 0651
266 0652
267 0653
268 0654
269 0655
270 0656
271 0657
272 0658
273 0659
274 0660
275 0661
276 0662
277 0663
278 0664
279 0665
280 0666
281 0667
282 0668
283 0669
284 0670
285 0671
286 0672
287 0673
288 0674
289 0675
290 0676
291 0677
292 0678
293 0679
294 0680
295 0681
296 0682
297 0683
298 0684

Conditional compile some debugging code.  If compiled with /VARIANT<>0, then if the
logical name SHOW$DEBUG is defined we will dump the lock state

IF %VARIANT NEQ 0
THEN
    If the logical name SHOW$DEBUG is defined, dump everything we found
    IF (BEGIN
        LOCAL
            lcl_buf : VECTOR [256, BYTE],
            out_dsc : VECTOR [2, LONG];
            out_dsc[0] = 256;
            out_dsc[1] = lcl_buf;
            ($trnlog (lognam=%ASCII 'SHOW$DEBUG', rslten=out_dsc, rslbuf=out_dsc) EQL ss$_normal)
        END)
    THEN
        BEGIN
            IF NOT .lclbuf[lcl_enq_status]
            THEN
                SIGNAL(show$_lockerr, 2, .scratch[d_b_devlen], scratch[d_t_device], .lclbuf[lcl_enq_status]);
                show$write_line (%ASCII ' - Lock !XL, length of items !XL, $enq status !XL, value block !XL !XL
                                lclbuf[0,0,32,0]);
                show$write_line (%ASCII ' - Mounted on nodes: !AF', arglist);
                show$write_line (%ASCII ' - Lock id PID CSID Remlkid Remcsid
                                arglist);
                incr k from 0 to .lclbuf[lcl_ret_length]-1 by lki$_length
                do
                    begin
                        local
                            nodename : VECTOR [16, BYTE],
                            nodename2 : VECTOR [16, BYTE];
                        bind
                            lki = lokbuf[k,0,32,0] : $BBLOCK;
                        arglist[0] = .lki[lki$_lockid];
                        arglist[1] = .lki[lki$_pid];
                        arglist[2] = .lki[lki$_sysid];
                        arglist[3] = 0;
                        arglist[4] = nodename;
                        arglist[5] = .lki[lki$_remlkid];
                        arglist[6] = .lki[lki$_remsysid];
                        arglist[7] = 0;
                        arglist[8] = nodename2;
                        arglist[9] = .lki[lki$_rqmode];
                        arglist[10] = .lki[lki$_grmode];
                        arglist[11] = .lki[lki$_queue];
                        IF .lki[lki$_sysid] NEQ 0
                        THEN
                            BEGIN
                                LOCAL
                                    itemlist : VECTOR [4, LONG];
                                    itemlist[0] = (syi$_nodename^16 OR 16);
                                    itemlist[1] = nodename;
                                    itemlist[2] = arglist[3];
                                    itemlist[3] = 0;
```

```
299      U 0685      $getsyi (csidadr=lki[lki$l_sysid],itmlst=itemlist);
300      U 0686      END;
301      U 0687      IF .lki[lki$l_remsysid] NEQ 0
302      U 0688      THEN
303      U 0689      BEGIN
304      U 0690      LOCAL
305      U 0691      itemlist : VECTOR [4, LONG];
306      U 0692      itemlist[0] = (syi$nodename^16 OR 16);
307      U 0693      itemlist[1] = nodename2;
308      U 0694      itemlist[2] = arglist[7];
309      U 0695      itemlist[3] = 0;
310      U 0696      $getsyi (csidadr=lki[lki$l_remsysid],itmlst=itemlist);
311      U 0697      END;
312      U 0698      show$write_line (
313      U 0699      %ASCII ' - !XL !XL !XL !8<(!AF)!> !XL !XL !8<(!AF)!> !XB !XB !XB', arglis
314      U 0700      end;
315      U 0701      !
316      U 0702      ! Format the buffer, 32 bytes at a time
317      U 0703      !
318      U 0704      show$write_line (%ASCII ' - Formatted dump of LKIS_LOCKS buffer:', arglist);
319      U 0705      incr k from 0 to .lclbuf[lcl_ret_length]-1 by 32
320      U 0706      do
321      U 0707      begin
322      U 0708      !
323      U 0709      ! Move the next chunk of data to the intermediate buffer
324      U 0710      !
325      U 0711      arglist[0]=.lokbuf[.k+28,0,32,0];
326      U 0712      arglist[1]=.lokbuf[.k+24,0,32,0];
327      U 0713      arglist[2]=.lokbuf[.k+20,0,32,0];
328      U 0714      arglist[3]=.lokbuf[.k+16,0,32,0];
329      U 0715      arglist[4]=.lokbuf[.k+12,0,32,0];
330      U 0716      arglist[5]=.lokbuf[.k+8,0,32,0];
331      U 0717      arglist[6]=.lokbuf[.k+4,0,32,0];
332      U 0718      arglist[7]=.lokbuf[.k,0,32,0];
333      U 0719      arglist[8]=32;
334      U 0720      arglist[9]=lokbuf[.k,0,32,0];
335      U 0721      arglist[10]=.k;
336      U 0722      show$write_line (%ASCII ' - !8(9XL) !32AF !XW', arglist);
337      U 0723      end;
338      U 0724      END;
339      U 0725      %FI      ! End of variant for debug listing
340      U 0726      !
341      U 0727      RETURN;
342      U 0728      END;
```

.TITLE SHODEVCLU  
.IDENT \V04-000\

.PSECT \$PLITS\$,NOWRT,NOEXE,2

29 74 6E 6D 20 65 74 6F 6D 65 72 28 00000 P.AAA: .ASCII \(\remote mnt)\ :

.PSECT \$OWNS\$,NOEXE,2

00000000 00000 LOCAL\_CSID:  
.LONG 0 :



				OFFC 00000		
			SE	FAF4	CE	9E 00002
			5A	08	AC	D0 00007
					6A	D4 0000B
			57	04	AC	D0 0000D
			59	70	A7	9E 00011
			01		13	EF 00015
04	50	69	06		50	F0 0001A
				0000	CF	D5 00020
					31	12 00024
			6E	10D00004	8F	D0 00026
	04		AE	0000	CF	9E 0002D
				08	AE	7C 00033
					7E	7C 00036
					7E	D4 00038
				0C	AE	9F 0003A
					7E	7C 0003D
					7E	D4 0003F
00000000G			00		07	FB 00041
			52		50	D0 00048
			09		52	E8 0004B
					52	DD 0004E
00000000G			00		01	FB 00050
			6E		04	D0 00057 1\$:
	04		AE		57	D0 0005A
	08		AE	E4	AD	9E 0005E
	0C		AE	04B0	8F	3C 00063
	10		AE	40	AE	9E 00069
					5E	DD 0006E
				0000V	CF	9F 00070
00000000G			00		02	FB 00074
			52		50	D0 0007B
			45		52	E8 0007E
			50	08	A7	9E 00081
			0C		52	D1 00085
					26	12 00088
					7E	D4 0008A
			7E	00000000G	00	7D 0008C
			7E	00000000G	00	7D 00093
					05	BB 0009A
			7E	06	A7	9A 0009C
					02	DD 000A0
				00000000G	8F	DD 000A2
00000000G			00		0A	FB 000A8
					04	000AF
					05	BB 000B0 2\$:
			7E	06	A7	9A 000B2
					02	DD 000B6

  

		.EXTRN	SHOW\$WRITE LINE	
		.EXTRN	SHOW\$ LOCKERR, LCK\$GL MAXID	
		.EXTRN	LCK\$GL IDTBL, KERNEL ACCVIO	
		.EXTRN	SYSS\$GETSYI, SYSS\$CMKRNL	
		.PSECT	SCODES, NOWRT, 2	
		.ENTRY	SCAN CLUSTER LOCKS, Save R2,R3,R4,R5,R6,R7,-;	0514
		MOVAB	R8,R9,R10,R11	
		MOVL	-1292(SP), SP	
		CLRL	BUFFER, R10	0542
		MOVL	(R10)	0548
		MOVAB	SCRATCH, R7	0549
		EXTZV	112(R7), R9	
		INSV	#19, #1, (R9), R0	
		TSTL	R0, #6, #1, 4(R7)	
		BNEQ	LOCAL_CSID	0554
		MOVL	1\$	
		MOVAB	#282066948, ARGLIST	0557
		CLRQ	LOCAL_CSID, ARGLIST+4	0558
		CLRQ	ARGLIST+8	0559
		CLRL	-(SP)	0560
		PUSHAB	-(SP)	
		CLRQ	ARGLIST	
		CLRL	-(SP)	
		CALLS	-(SP)	
		MOVL	#7, SYSS\$GETSYI	
		BLBS	R0, STATUS	
		PUSHL	STATUS, 1\$	0562
		CALLS	STATUS	
		MOVL	#1, LIB\$STOP	0568
		MOVL	#4, ARGLIST	0569
		MOVAB	R7, ARGLIST+4	0570
		MOVZWL	LCLBUF, ARGLIST+8	0571
		MOVAB	#1200, ARGLIST+12	0572
		PUSHL	LOKBUF, ARGLIST+16	0574
		PUSHAB	SP	
		CALLS	GET_LOCK_INFO	
		MOVL	#2, SYSS\$CMKRNL	
		BLBS	R0, STATUS	
		MOVAB	STATUS, 3\$	0578
		CMP	8(R7), R0	0577
		BNEQ	STATUS, #12	
		CLRL	2\$	
		MOVQ	-(SP)	0578
		MOVQ	KERNEL_ACCVIO+8, -(SP)	0579
		PUSHR	KERNEL_ACCVIO, -(SP)	
		MOVZBL	#*M<R0,R2>	0578
		PUSHL	6(R7), -(SP)	
		CALLS	#2	
		RET	#SHOW\$ LOCKERR	
		PUSHR	#10, LIB\$SIGNAL	
		MOVZBL	#*M<R0,R2>	0580
		PUSHL	6(R7), -(SP)	
			#2	

00000000G	00	00000000G	8F	DD	000B8	PUSHL	#SHOWS LOCKERR	...	
			05	FB	000BE	CALLS	#5, LIBSSIGNAL	...	0576
	5B	E8	AD	3C	000C5	RET		...	0588
			5B	D7	000CA	MOVZWL	LCLBUF+4, R11	...	
	58		18	CE	000CC	DECL	R11	...	0599
			63	11	000CF	MNEGL	#24, K	...	
	56	40	AE48	9E	000D1	BRB	6\$	...	0592
0000'	CF	14	A6	D1	000D6	MOVAB	LOKBUF[K], R6	...	0599
			56	13	000DC	CMPL	20(R6), LOCAL_CSID	...	
		0D	A6	95	000DE	BEQL	6\$	...	0601
			51	13	000E1	TSTB	13(R6)	...	
			6A	D6	000E3	BEQL	6\$	...	0604
	50		6A	D0	000E5	INCL	(R10)	...	0605
	6A40	14	A6	D0	000E8	MOVL	(R10), R0	...	
	34	F0	AD	E9	000ED	MOVL	20(R6), (R10)[R0]	...	0606
04	A7	80	8F	88	000F1	BLBC	LCLBUF+12, 5\$	...	0609
02	A9		08	88	000F6	BISB2	#128, 4(R7)	...	0610
		00CC	C7	B6	000FA	BISB2	#8, 2(R9)	...	0611
03	50	F0	AD	01	EF	INCW	204(R7)	...	0613
	A9		01	50	F0	EXTZV	#1, #1, LCLBUF+12, R0	...	
	50	F0	AD	01	04	INSV	R0, #0, #1, 3(R9)	...	0615
				50	EF	EXTZV	#4, #1, LCLBUF+12, R0	...	
69				50	D2	MCOML	R0, R0	...	
				50	F0	INSV	R0, #25, #1, (R9)	...	0616
		04	A7	06	E0	BBS	#6, 4(R7), 5\$	...	0618
	00B8	C7	0000'	0C	28	MOV3	#12, P.AAA, 184(R7)	...	0620
			05	A6	91	CMPB	13(R6), #5	...	
		0D		09	12	BNEQ	6\$	...	0623
	02	A9	80	8F	88	BISB2	#128, 2(R9)	...	0624
	05	A7		01	88	BISB2	#1, 5(R7)	...	0588
FF97		58		5B	F1	ACBL	R11, #24, K, 4\$	...	0728
				04	0013A	RET		...	

; Routine Size: 315 bytes, Routine Base: \$CODE\$ + 0000



```

344 0729 1 GLOBAL ROUTINE get_lock_info (scratch : REF $BBLOCK, lclbuf : REF $BBLOCK, lokbuf_size, lokbuf : REF $BBLOCK
345 0730 BEGIN
346 0731
347 0732 ---
348 0733
349 0734 This routine is called in KERNEL mode to scan the device lock data base and
350 0735 determine any cluster-wide information which is available.
351 0736
352 0737 Inputs
353 0738     SCRATCH - address of scratch data for this device
354 0739     LOKBUF_SIZE - size of lock info buffer
355 0740     LOKBUF - lock info buffer for the $GETLKI call, passed in so that we
356 0741             don't have kernel stack restrictions on the size of the buffer
357 0742
358 0743 Outputs
359 0744     SCRATCH - some values "adjusted" for cluster-wide information
360 0745     LCLBUF - output lock information for control and debug listings
361 0746     LOKBUF - lock info vector
362 0747
363 0748 ---
364 0749
365 0750 LOCAL
366 0751     iosb : $BBLOCK [8],
367 0752     itemlist : $BBLOCK [16],
368 0753     lokbuf_len : $BBLOCK [4],
369 0754     lksb : $BBLOCK [24],
370 0755     name : VECTOR [20, BYTE],
371 0756     name_desc : VECTOR [2, LONG],
372 0757     status;
373 0758
374 0759 !
375 0760 ! Trap anything weird, and turn it into a return
376 0761
377 0762 ENABLE
378 0763     get_lock_info_handler;
379 0764
380 0765 !
381 0766 ! Get a null-mode lock on the device name
382 0767
383 0768 (name[0]) = 'SYS$';
384 0769 CH$MOVE (.scratch[d_b_devlen], scratch[d_t_device], name[4]);
385 0770 name_desc[0] = 4 + .scratch[d_b_devlen];
386 0771 name_desc[1] = name;
387 0772 status = $ENQW (efn=0,
388 0773                lkmode=LCK$K_NLMODE,
389 0774                lksb=lksb,
390 0775                flags=(LCK$M_NOQUEUE OR LCK$M_VALBLK OR LCK$M_SYNCSTS OR LCK$M_SYSTEM),
391 0776                resnam=name_desc,
392 0777                acmode=0);
393 0778 lclbuf[lcl null lkid]=lksb[4,0,32,0];
394 0779 CH$MOVE (16,lksb[8,0,32,0],lclbuf[lcl_val_block]);
395 0780 IF .status
396 0781 THEN status = .lksb[0,0,16,0];
397 0782 lclbuf[lcl_eng_status] = .status;
398 0783 IF NOT .status
399 0784 THEN
400 0785     IF .status NEQ ss$_valnotvalid
```

! status block + value block

! Stick the null lock id into the buffer

! Copy the value block

! If the enqueue worked then check the stat block

! Save \$enq status

! Cope with a value block not being valid

```

: 401      0786      2      THEN
: 402      0787      BEGIN
: 403      0788      $DEQ (lkid=.lksb[4,0,32,0]);      ! Release the lock, just in case one was granted
: 404      0789      RETURN .status;      ! Return with status
: 405      0790      END;
: 406      0791
: 407      0792
: 408      0793      ! Get lock information, and find out all the other locks on this resource name
: 409      0794
: 410      0795      itemlist[0,0,16,0] = .lokbuf size;      ! Size of scratch buffer
: 411      0796      itemlist[2,0,16,0] = LKIS LOCKS;      ! Find out which other locks
: 412      0797      itemlist[4,0,32,0] = .lokbuf;      ! Address of scratch buffer
: 413      0798      itemlist[8,0,32,0] = .lokbuf_len;      ! Address for returned length
: 414      0799      itemlist[12,0,32,0] = 0;      ! End of itemlist
: 415      0800      lokbuf_len = 0;      ! Zero returned length (?? bogus when GETLKI works)
: 416      0801      status = $GETLKI (efn=0,
: 417      0802      lkidadr=lksb[4,0,32,0],
: 418      0803      itmlst=itemlist,
: 419      0804      iosb=iosb);
: 420      0805      lclbuf[lcl_lengths] = .lokbuf_len[0,0,32,0];      ! Save both length fields
: 421      0806      IF .status      ! If the getlki worked then check the stat block
: 422      0807      THEN status = .iosb[0,0,16,0];
: 423      0808
: 424      0809      ! Release the null lock
: 425      0810
: 426      0811      $DEQ (lkid=.lksb[4,0,32,0]);
: 427      0812
: 428      0813
: 429      0814      RETURN .status;      ! Return with status
: 430      0815      1      END;
```

P  
P  
P

10 AE

```

                                00FC 00000
                                AE 9E 00002
                                CF DE 00006
                                0C AE 24535953 8F DO 0000B
                                56 04 AC DO 00013
                                50 06 A6 9A 00017
                                08 A6 50 28 0001B
                                04 AE 06 A6 9A 00021
                                04 AE 04 C0 00026
                                08 AE 0C AE 9E 0002A
                                7E 7C 0002F
                                7E 7C 00031
                                7E 7C 00033
                                1C AE 9F 00035
                                1D DD 00038
                                40 AE 9F 0003A
                                7E 7C 0003D
                                00000000G 00 0B FB 0003F
                                57 50 DO 00046
                                56 08 AC DO 00049
                                66 24 AE DO 0004D
```

```

.EXTRN SYSS$ENQW, SYSS$DEQ
.EXTRN SYSS$GETLKI
```

```

.ENTRY GET_LOCK_INFO, Save R2,R3,R4,R5,R6,R7
MOVAB -80(SP), SP
MOVAL 4$, (FP)
MOVL #609442131, NAME
MOVL SCRATCH, R6
MOVZBL 6(R6), R0
MOVCL3 R0, 8(R6), NAME+4
MOVZBL 6(R6), NAME_DESC
ADDL2 #4, NAME_DESC
MOVAB NAME, NAME_DESC+4
CLRQ -(SP)
CLRQ -(SP)
CLRQ -(SP)
PUSHAB NAME_DESC
PUSHL #29
PUSHAB LKSB
CLRQ -(SP)
CALLS #11, SYSS$ENQW
MOVL R0, STATUS
MOVL LCLBUF, R6
MOVL LKSB+4, (R6)
```

```

: 0729
: 0730
: 0768
: 0769
: 0770
: 0771
: 0777
: 0778
```



0C	A6	28	AE	10	28	00051	MOV C3	#16, LKSB+8, 12(R6)	:	0779	
			04	57	E9	00057	BLBC	STATUS, 1\$	:	0780	
			57	20	AE	3C	MOVZWL	LKSB, STATUS	:	0781	
	08		A6	57	D0	0005A	1\$:	MOV L	STATUS, 8(R6)	:	0782
			09	57	E8	00062	BLBS	STATUS, 2\$	:	0783	
	000009F0		8F	57	D1	00065	CMPL	STATUS, #2544	:	0785	
				3D	12	0006C	BNEQ	3\$	:		
	38		AE	0C	AC	B0	2\$:	MOVW	LOKBUF SIZE, ITEMLIST	:	0795
	3A		AE	0208	8F	B0	MOVW	#520, ITEMLIST+2	:	0796	
	3C		AE	10	AC	D0	MOV L	LOKBUF, ITEMLIST+4	:	0797	
	40		AE		6E	9E	MOVAB	LOKBUF LEN, ITEMLIST+8	:	0798	
				44	AE	D4	CLRL	ITEMLIST+12	:	0799	
					6E	D4	CLRL	LOKBUF_LEN	:	0800	
					7E	7C	CLRQ	-(SP)	:	0804	
					7E	D4	CLRL	-(SP)	:		
				54	AE	9F	PUSHAB	IOSB	:		
				48	AE	9F	PUSHAB	ITEMLIST	:		
				38	AE	9F	PUSHAB	LKSB+4	:		
					7E	D4	CLRL	-(SP)	:		
	00000000G		00		07	FB	CALLS	#7, SYS\$GETLKI	:		
			57		50	D0	MOV L	RO, STATUS	:		
	04		A6		6E	D0	MOV L	LOKBUF_LEN, 4(R6)	:	0805	
			04		57	E9	BLBC	STATUS, 3\$	:	0806	
			57		48	AE	MOVZWL	IOSB, STATUS	:	0807	
						7E	CLRQ	-(SP)	:	0812	
						7E	CLRL	-(SP)	:		
				30	AE	DD	PUSHL	LKSB+4	:		
	00000000G		00		04	FB	CALLS	#4, SYS\$DEQ	:		
			50		57	D0	MOV L	STATUS, RO	:	0814	
						04	RET		:	0815	
					0000	000BD	4\$:	.WORD	Save nothing	:	0730
					7E	D4	CLRL	-(SP)	:		
					5E	DD	PUSHL	SP	:		
			7E		AC	7D	MOVQ	4(AP), -(SP)	:		
	0000V		CF		03	FB	CALLS	#3, GET_LOCK_INFO_HANDLER	:		
					04	000CC	RET		:		

; Routine Size: 205 bytes, Routine Base: \$CODE\$ + 013B



.EXTRN	LIB\$\$SIG_TO_RET	
.ENTRY	GET_LOCK_INFO_HANDLER, Save R2,R3,R4,R5,R6	: 0816
MOVL	SIG, R6	: 0839
CMPL	4(R6), #12	:
BNEQ	1\$	:
MOVCL	#16, 8(R6), KERNEL_ACCVIO	: 0842
PUSHL	MECH	: 0843
PUSHL	R6	:
CALLS	#2, LIB\$\$SIG_TO_RET	:
RET		:
MOVZWL	#2328, R0	: 0846
RET		: 0847

; Routine Size: 40 bytes, Routine Base: \$CODES + 0208



: 465 0848 1 END  
: 466 0849 0 ELUDOM

.EXTRN LIB\$SIGNAL, LIB\$STOP

## PSECT SUMMARY

Name	Bytes	Attributes
\$OWNS	4	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$SPLITS	12	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$CODES	560	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)

## Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	40	0	1000	00:01.9

## COMMAND QUALIFIERS

: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:SHODEVCLU/OBJ=OBJ\$:SHODEVCLU MSRC\$:SHODEVCLU/UPDATE=(ENH\$:SHODEVCLU)

: Size: 560 code + 16 data bytes  
: Run Time: 00:22.9  
: Elapsed Time: 01:13.1  
: Lines/CPU Min: 2227  
: Lexemes/CPU-Min: 57993  
: Memory Used: 161 pages  
: Compilation Complete



0055 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

